



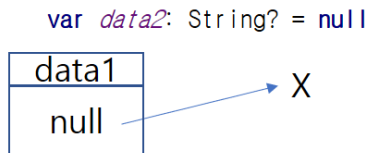
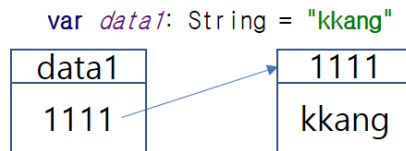
## 15장. Null 안전성과 예외처리



# 15.1. Null 안전성

## 15.1.1. Null 안전성이란?

- Null 이란 프로그램에서 값이 아무것도 대입되지 않은 상태



- The Billion Dollar Mistake
- Null-Safety 란 Null에 다양한 처리를 도와줌으로서 Null에 의한 NPE이 발생하지 않는 프로그램의 작성을 작성할수 있게 해준다는 개념

# 15.1. Null 안전성

## 15.1.2. Null 허용과 Null 불허

- 프로퍼티의 타입을 Nullable과 Non-Nullable로 구분

```
var data1: String = "kkang"

var data2: String? = null

fun main(args: Array<String>) {
    data1=null//error
}
```

```
var data1: String = "kkang"

var data2: String? = null

fun myFun(arg: String){
}

fun main(args: Array<String>) {
    data2="hello"

    val data3: String? = data1
    val data4: String=data2//error

    myFun(data2)//error
}
```

# 15.1. Null 안전성

## 15.1.3. Null 확인 연산자 ?.

- Null 체크

```
fun main(args: Array<String>) {  
    var data1: String? = "kkang"  
  
    val length1: Int? = if(data1 != null){  
        data1.length  
    } else {  
        null  
    }  
  
}
```

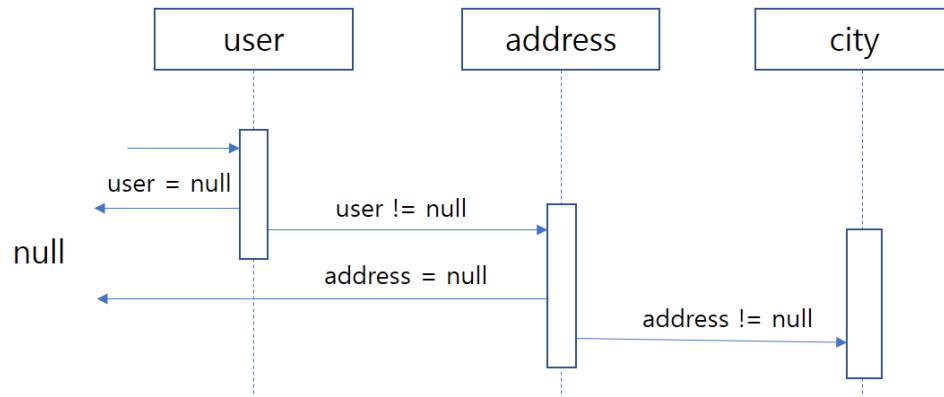
```
fun main(args: Array<String>) {  
    var data1: String? = "kkang"  
  
    var length2: Int? = data1?.length  
    println(length2)  
  
    data1 = null  
    length2 = data1?.length  
    println(length2)  
  
}
```

## 15.1. Null 안전성

- Null 체크는 객체의 연결 구조에서도 사용이 가능

```
class Address {  
    val city: String?="seoul"  
}  
  
class User {  
    val address: Address? = Address()  
}  
  
fun main(args: Array<String>) {  
    val user: User? = User()  
  
    println(user?.address?.city)  
}
```

user?.address?.city



## 15.1. Null 안전성

- Null 이 아닌 경우 특정 구문이 수행되어야 하는 경우

```
fun main(args: Array<String>) {  
    val array= arrayOf("hello", null, "kkang")  
  
    array.forEach {  
        if(it != null){  
            println("$it .. ${it.length}")  
        }  
    }  
  
    array.forEach {  
        it?.let {  
            println("$it .. ${it.length}")  
        }  
    }  
}
```

## 15.1. Null 안전성

### 15.1.4. 엘비스 연산자 ?:

- Null 인 경우에 처리를 명시

```
fun main(args: Array<String>) {  
    var data: String? = "kkang"  
  
    var length: Int = if(data != null){  
        data.length  
    }else {  
        -1  
    }  
  
    data=null  
  
    length=data?.length ?: -1  
  
    println(length)  
  
    data ?: println("data is null")  
}
```

### 실행결과

-1

data is null

## 15.1. Null 안전성

### 15.1.5. 예외 발생 연산자 !!

- Null 인경우 Exception 을 발생시키기 위한 연산자

```
fun main(args: Array<String>) {  
    var data: String? = "kkang"  
  
    data!!.length  
  
    data=null  
  
    data!!.length  
}
```

### 실행결과

Exception in thread "main" kotlin.KotlinNullPointerException



## 15.1. Null 안전성

### 15.1.6. 안전한 캐스팅

- `as` 연산자 이용시 캐스팅이 불가능한 경우는 `ClassCastException` 이 발생
- `as?` 연산자는 `ClassCastException` 이 발생해야 하는 상황에 에러 발생 없이 `Null` 이 리턴

```
fun main(args: Array<String>) {  
    val strData : String = "kkang"  
  
    val intData: Int? = strData as? Int  
  
    println(intData)  
}
```

## 15.2. 예외처리

### 15.2.1. try-catch-finally 구문으로 예외처리

```
fun main(args: Array<String>) {    try {  
    println("try top...")  
  
    val data: String = "10"  
  
    val intData: Int? = data.toInt()  
  
    println("try bottom...")  
} catch (e: Exception){  
    println("catch.....")  
  
} finally {  
    println("finally....")  
}  
}
```

#### 실행결과

```
try top...  
try bottom...  
finally....
```

## 15.2. 예외처리

```
fun main(args: Array<String>) {  
    try {  
        println("try top...")  
  
        val data: String = "kkang"  
  
        val intData: Int? = data.toInt()  
  
        println("try bottom...")  
    } catch (e: Exception){  
        println("catch.....${e.toString()}")  
  
    } finally {  
        println("finally....")  
    }  
}
```

### 실행결과

try top...

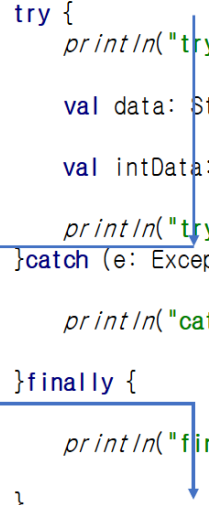
catch.....java.lang.NumberFormatException: For input string: "kkang"

finally....

## 15.2. 예외처리

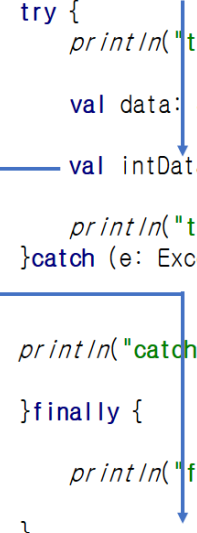
Exception 이 발생하지 않은 경우

```
try {  
    println("try top...")  
  
    val data: String = "kkang"  
  
    val intData: Int? = data.toInt()  
  
    println("try bottom...")  
} catch (e: Exception){  
    println("catch.....")  
} finally {  
    println("finally....")  
}
```



Exception 이 발생한 경우

```
try {  
    println("try top...")  
  
    val data: String = "kkang"  
  
    val intData: Int? = data.toInt()  
  
    println("try bottom...")  
} catch (e: Exception){  
    println("catch.....${e.toString()}")  
} finally {  
    println("finally....")  
}
```



## 15.2. 예외처리

```
fun main(args: Array<String>) {  
    try{  
    }catch (e: Exception){  
    }  
  
    try {  
    }finally {  
    }  
  
    try {  
    }catch (e: Exception){  
    }catch (e: Exception){  
    }  
}
```

## 15.2. 예외처리

catch 가 여러 개 정의된 경우

```
fun some(array: Array<Any>){
    try {
        println("try top...")
        val intData: Int= array[0] as Int
        val data: String = array[2] as String
        val data2: Int = data.toInt()
    }catch (e: ClassCastException){
        println("catch... ClassCastException")
    }catch (e: ArrayIndexOutOfBoundsException){
        println("catch... ArrayIndexOutOfBoundsException")
    }catch (e: Exception){
        println("catch... Exception... ${e.toString()}")
    }
}

fun main(args: Array<String>) {
    //cast exception
    val array= arrayOf("0", 1, "6")
    some(array);
    //index out of bound exception
    val array2= arrayOf(10,"5")
    some(array2)
    //Number format exception
    val array3=arrayOf(10, 0, "world")
    some(array3)
}
```

### 실행결과

```
try top...
catch... ClassCastException
try top...
catch... ArrayIndexOutOfBoundsException
try top...
catch... Exception... java.lang.NumberFormatException: For input string: "w
orld"
```

## 15.2. 예외처리

- try – catch 구문은 표현식으로도 사용이 가능
- 각 영역의 마지막 줄이 리턴값
- finally 부분은 표현식으로 이용되지는 않는다.

```
fun some1(arg: String): Int {  
    val parseData: Int = try {  
        println("try..top")  
        arg.toInt()  
    } catch (e: Exception){  
        println("${e.toString()}")  
        0  
    } finally {  
        println("finally...")  
        100  
    }  
    return parseData  
}  
fun main(args: Array<String>) {  
    println("${some1("10")}")  
  
    println("${some1("a")}")  
}
```

### 실행결과

```
try..top  
finally...  
10  
try..top  
java.lang.NumberFormatException: For input string: "a"  
finally...  
0
```

## 15.2. 예외처리

### 15.2.2. 예외 발생시키기

```
fun some(arg: Int): Int{
    if(arg < 1)
        throw Exception("parameter must be greater than zero")
    else {
        var sum=0
        for(i in 1..arg){
            sum += i
        }
        return sum
    }
}

fun main(args: Array<String>) {
    try {
        println("${some(5)}")

        println("${some(-1)}")

        println("main bottom....")
    } catch (e: Exception){
        println("Exception.... ${e.toString()}")
    }
}
```



## 15.2. 예외처리

```
fun some(arg: Int): Int{
    if(arg<1)
        throw Exception("parameter must be greater than zero")
    else {
        var sum=0
        for(i in 1..arg){
            sum += i
        }
        return sum
    }
}

fun main(args: Array<String>) {
    try {
        println("${some(5)}")
        println("${some(-1)}")
        println("main bottom....")
    } catch (e: Exception){
        println("Exception.... ${e.toString()}")
    }
}
```

Exception 발생

catch 실행

## 15.2. 예외처리

- Exception 클래스 정의

```
class MyException(msg: String): Exception(msg){
    val errorData: String = "some error data"
    fun errorFun(){
        println("errorFun call...")
    }
}

fun some1(){
    throw MyException("My Error...")
}

fun main(args: Array<String>) {
    try {
        some1()
    } catch (e: MyException){
        println("error message : ${e.toString()}")
        println("error data : ${e.errorData}")
        e.errorFun()
    }
}
```

## 15.2. 예외처리

- throw는 표현식

```
fun some1() {  
    val name: String? = null  
    val s: String = name ?: throw IllegalArgumentException("Name required")  
    println("some1 bottom")  
}  
  
fun some2(arg: Int): Nothing {  
    if(arg > 0)  
        throw Exception("some2 exception.. arg>0 true")  
    else  
        throw Exception("some2 exception.. arg>0 false")  
}  
  
fun main(args: Array<String>) {  
    try {  
        some1()  
    } catch (e: Exception) {  
        println(e.toString())  
    }  
  
    try {  
        some2(10)  
    } catch (e: Exception) {  
        println(e.toString())  
    }  
}
```

### 실행결과

```
java.lang.IllegalArgumentException: Name required  
java.lang.Exception: some2 exception.. arg>0 true
```